CS533: Intelligent Agents and Decision Making

Assignment 4: Reinforcement Learning for Optimal Parking

You can use any programming language for the project, including MATLAB. In addition to the write-up materials indicated below, you should also send all of your code when you submit the assignment. Submit the project through Canvas.

In this project you will implement, evaluate, and compare several reinforcement learning algorithm in the context of the "parking" MDP designed in assignment 3.

Part I: Create an MDP Simulator

In order to run RL experiments, you need to be able to simulate the RL agent taking actions in an MDP.

Your job in this part of the assignment is to create a general-purpose simulator that can be given the description of any MDP, using the MDP format from Assignment 3. There is no need to provide a nice user interface. The most important functionality is that the simulator provide an easy interface to both learning and non-learning agents that allows an agent to run many episodes and observe the states and rewards during those episodes. Your simulator should be able to take as input, a terminal state id (not all problems will have terminal states). The simulator will not do anything once when in a terminal state (zero reward and no state transitions).

PART II: Policy Simulation

Here you will test the simulator by using it to evaluate policies on MDPs (i.e. using the simulator to estimate the value function of a policy for an initial state).

Part a. Create a simple MPD, called MDP-test, which could have as few as 5 states. The MDP should have the following properties: 1) The optimal policy is obvious, 2) There are sub-optimal policies (i.e. not all actions are equally good), 3) It is easy to determine the value of a policy that selects random actions (i.e. always explores), 4) Some of the state transitions should be probabilistic (i.e. not a purely deterministic MDP). You should designate a particular state as the start state of the MDP, which is where all episodes will begin.

Describe your MDP and provide the file for it. State your estimate of the value of a random policy from the start state (for a selected discount factor) and the value of the optimal policy from that state. Now use your simulator to evaluate a policy that selects random actions. To do this, you simply need to simulate a number of episodes (say 100) of the policy, where you will want to put a maximum limit on the length of an episode. For each episode, which starts in the start state, you can accumulate the discounted total reward. The average of the discounted total reward across the episodes is the value estimate for the policy from the start state. Report the value that you measure and compare it to the value that you determined on your own.

Part b. In Assignment 3 you create two versions of the parking domain. Here you will estimate the value of several policies on those parking domain MDPs. (The description of this MDP is replicated at the end of this assignment for your convenience.)

1. Measure the performance of a random policy on the two MDPs. This policy should select the PARK action with some probability p and DRIVE with probability $1 - p$. You can specify p.

2. Measure the performance of the policy in both MDPs that DRIVES whenever O is TRUE and when O is FALSE selects PARK with probability p and otherwise DRIVES with probability $1 - p$.

3. Try to write your own SIMPLE policies (one for each MDP) that improves on the ones above and evaluate them. You do not need to work out "optimal policies", just implement one or two obvious improvements of the above.

Note that the in each case we are interested in having you report the value of the policy from the start state.

PART III: Reinforcement Learning

Implement three model-free reinforcement learning agents: Mone-Carlo On-Policy Control, SARSA, and Q-Learning. The agents should have a clean interface to your simulator so that they could be run on any MDP. For this assignment you should always use ϵ-greedy exploration and use a constant learning rate α. For the Monte-Carlo On-Policy Control you may want to follow the pseudo-code in Section 5.4 of Sutton & Barto, which has a more efficient implementation than the simpler one described in the lecture slides.

Your goal is to measure and compare the performances of the agents as they learn in the 3 MDPs that were described above (the small one you designed + two parking MDPs). Select a single value of the learning rate α to use that appears to work reasonably based on initial experimentation and use that for all experiments. Repeat experiments using ϵ values of 0.05 and 0.3, which corresponds to moderate exploration and aggressive exploration. Overall each RL algorithm will be trained 6 times, twice for each of the three MDPs.

For the experimental evaluations and comparisons it is suggested that you allow the agents to learn for N trials and then measure the performance of the resulting greedy policy (measuring performance as in part II), then run another N trials of learning and measure the performance, continue this process of learning and evaluation until the performance does not improve. Importantly you should use a greedy policy (exploitation) during the evaluation trials. An exploration policy should only be used during learning. You will want to select N so that you can see some change in performance.

You should produce a BRIEF report about the results, including learning curves for the different agents. The report should attempt to compare the different algorithms and explain any differences observed and the impact of the ϵ parameter. You should compare the learned performance with that of the hand-coded policies above for the parking domain.

Note that you will want to limit the number of steps for each episode to avoid a exceedingly long episodes.

What You Need to Produce

1. Provide me with the code for you implementation of the simulator and non-learning and learning agent.

2. Provide a BRIEF writeup that gives:

   (a) Your choices in Part II

   (b) The performance of the policies in part II for each MDP (describe the policies that you designed)

(c) The brief description of results for part III as described above.

Submit this material through Canvas.

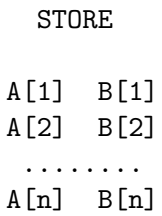
Appendix: Parking Domain MDP Description

NOTE: The MDP description is just like the one for assignment 3, just repeated here for your convenience. The key distinction for this project is that Part I asks you to design a simulator for the environment.

Most people like to park close to wherever they are going. This desire often seems to lead to irrational behavior—e.g. driving around a parking lot for several minutes looking for a slightly closer spot. In this project you will put a reinforcement learning agent in this situation and observe its behavior.

The first part of the project is to design simple MDPs to represent the experience of parking a car. The MDPs will be very similar, differing primarily in the exact parameter values you choose. You will then design a simple simulator for these environments that your reinforcement learning agent can act in. The MDP should capture the following qualitative characteristics of the parking problem:

1. Parking spots closer to the store are more desirable to the agent.
2. The probability that a parking spot is available is smaller the closer a spot is to the store.
3. It is undesirable to spend much time searching for a spot (i.e. driving around the parking lot).
4. The parking lot should have two rows of parking spaces A and B (see below figure) that must be traveled in a loop. These rows are parallel to each other and have n parking spots labeled $A[1], \ldots, A[n]$ and $B[1], \ldots, B[n]$. $A[1]$ and $B[1]$ are closest to the store, $A[n]$ and $B[n]$ are furthest from the store. In row A the agent can only drive toward the store (i.e. move from $A[i]$ to $A[i-1]$) unless the agent is in $A[1]$ where it can only move to $B[1]$. When the agent is in row B it can only move away from the store (i.e. move from $B[i]$ to $B[i+1]$), unless the agent is in row $B[n]$ where it can only move to $A[n]$. Thus the agent can only drive in a "circular" motion around rows A and B.



5. The parking spots closest to the store $A[1]$ and $B[1]$ are handicap spots. There is a high cost (negative reward) for parking in these spots (although they are desirable with respect their closeness to the store). Also the probability that the handicap spot is available should be high.
6. If an agent attempts to park in a spot that contains a car, then there is a high cost (negative reward) since there will be a collision.

7. When a parking trial begins the agent is randomly placed at either $B[1]$ or $A[n]$. A trial ends when the agent decides to park, resulting in either a collision or a parked car.

You need to specify an MDP that roughly represents the above features. Here is a suggested structure, but you are free to try something else.

- **State Space:** Each state is a triple (L, O, P) where L is a location (one of the $A[i]$ or $B[i]$), O is a boolean variable that is TRUE if the spot at that location is occupied by another car, and P is a boolean variable that is TRUE if the agent is parked (or tried to park) at the location. Thus there will be $8n$ states since there are $2n$ locations and two values of A and B. Initially P will be FALSE and the trial ends when P is true. That is, any state where P is true is considered to be a terminal state.

- **Actions:** There are three actions PARK, DRIVE. When the action DRIVE is taken the agent is moved to the next parking spot (according to the circular driving pattern described above) and a coin is flipped to set the value of O. The probability that O is true should increase for spots closer to the store (the details of this are up to you). However the probability that O is true for handicap spots $A[1]$ or $B[1]$ should be very high. The DRIVE action does not change the P variables (P is initialized to be FALSE). When the action PARK is taken the value of P is set to TRUE and L and O are left unchanged. Parking should result in a terminal state where the episode ends.

- **Reward:** States where P is FALSE will get a negative reward that represents the cost of driving. This way if the agent drives for a long time it will accumulate negative reward. Thus,long driving times will (eventually) look undesirable.

  For terminal states (any state where P is TRUE) the reward should be based on the location and the value of O. Clearly if O is TRUE then we want there to be a large cost, since this corresponds to a collision. If O is FALSE then we want the reward to based on two factors. There should be more reward for parking closer to the store, but parking in $A[1]$ or $B[1]$ (the two closest spots) should be discouraged by giving a smaller reward (since they are handicap spots).

- **Discounting:** You can use a discount factor of 1 (no discounting). However, it will be important to put a horizon limit on the trials, otherwise you might end up driving around the parking lot for a very long time during learning.

From this discussion the total reward over a trial is the sum of the number of driving steps plus the parking reward. An agent needs to learn how to balance the time spent searching for a better spot with the location of the spot. The rational balance will depend on the relative cost of driving versus parking further away.